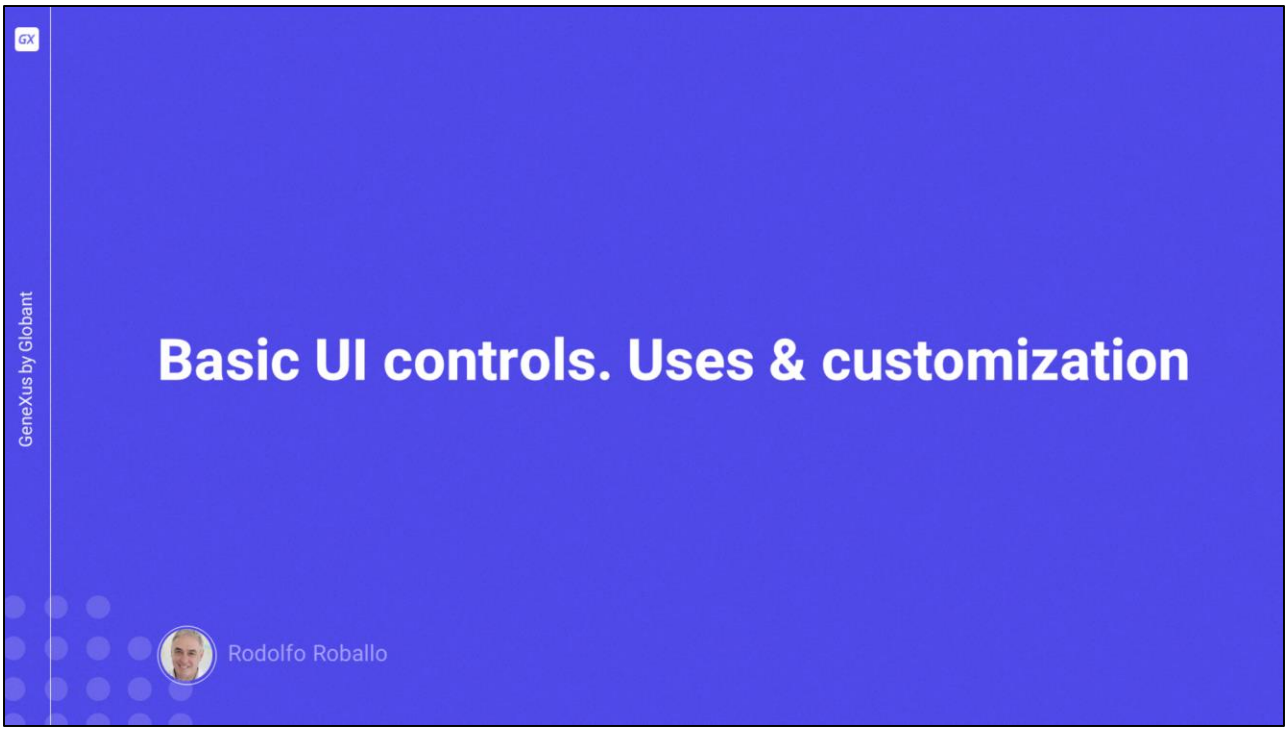
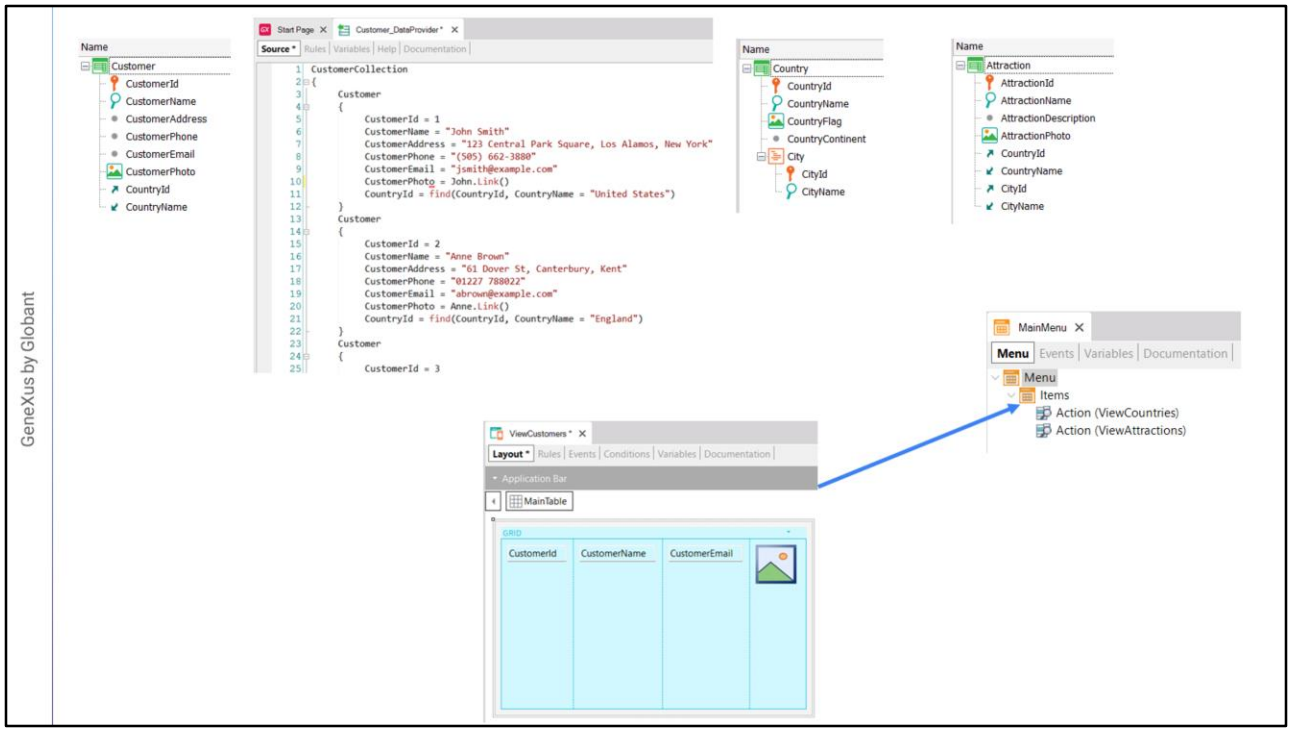


Algo que es fundamental en cualquier aplicación informática es el correcto uso de los controles de pantalla disponibles, para brindar la mejor experiencia de usuario.

En este video veremos los controles más usados, cómo podemos organizarlos en el layout del panel y personalizar su apariencia.

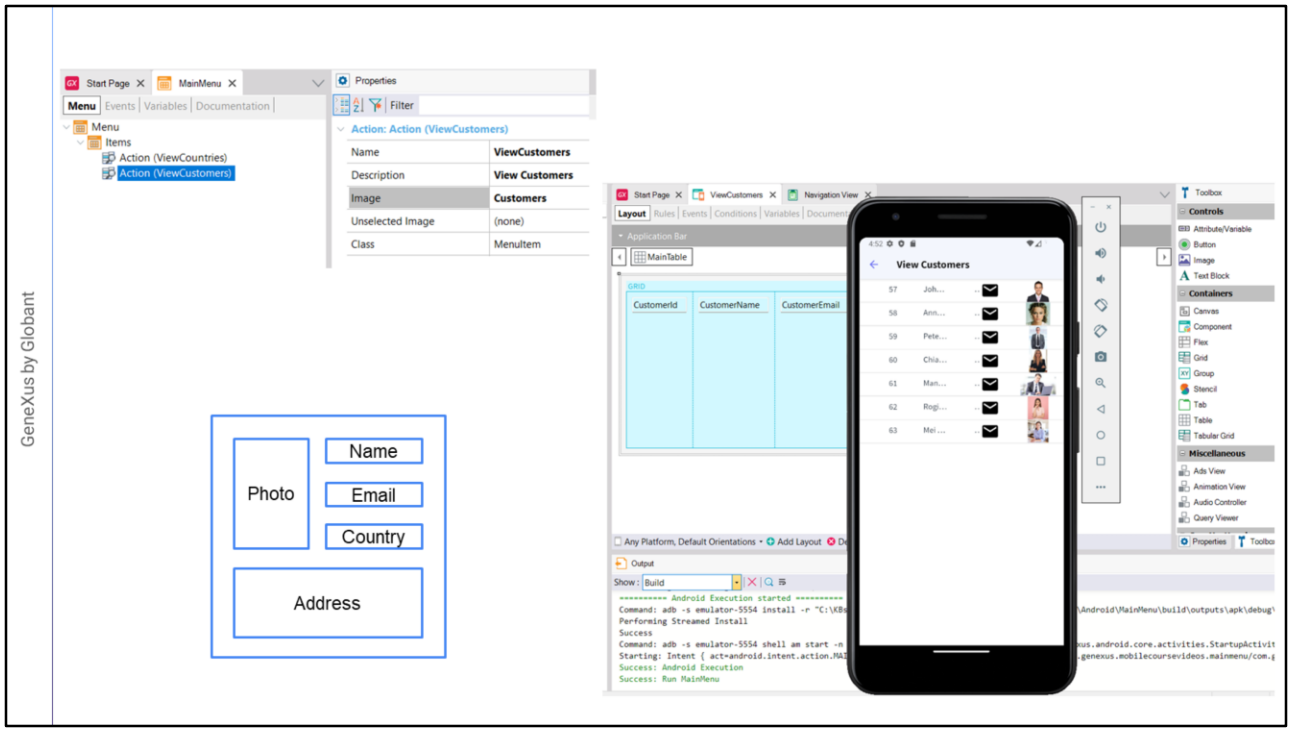


Vamos a crear un panel para ver los datos de los clientes de la agencia de viajes.

Primero que nada importamos a nuestra KB el archivo Customers.xpz que encontramos en la sección Materiales de la página del curso.

Vemos que se importó la transacción Customer con atributos para los datos principales como el identificador, nombre, dirección, teléfono, email y su foto. También vemos que se importó el data provider Customer_DataProvider que incluye los datos de los clientes y otros objetos más.

Para poder ver los datos de los clientes, creamos el panel ViewCustomers, arrastramos un control grid con los atributos CustomerId, CustomerName, CustomerEmail, CustomerPhoto y CountryName...y salvamos.



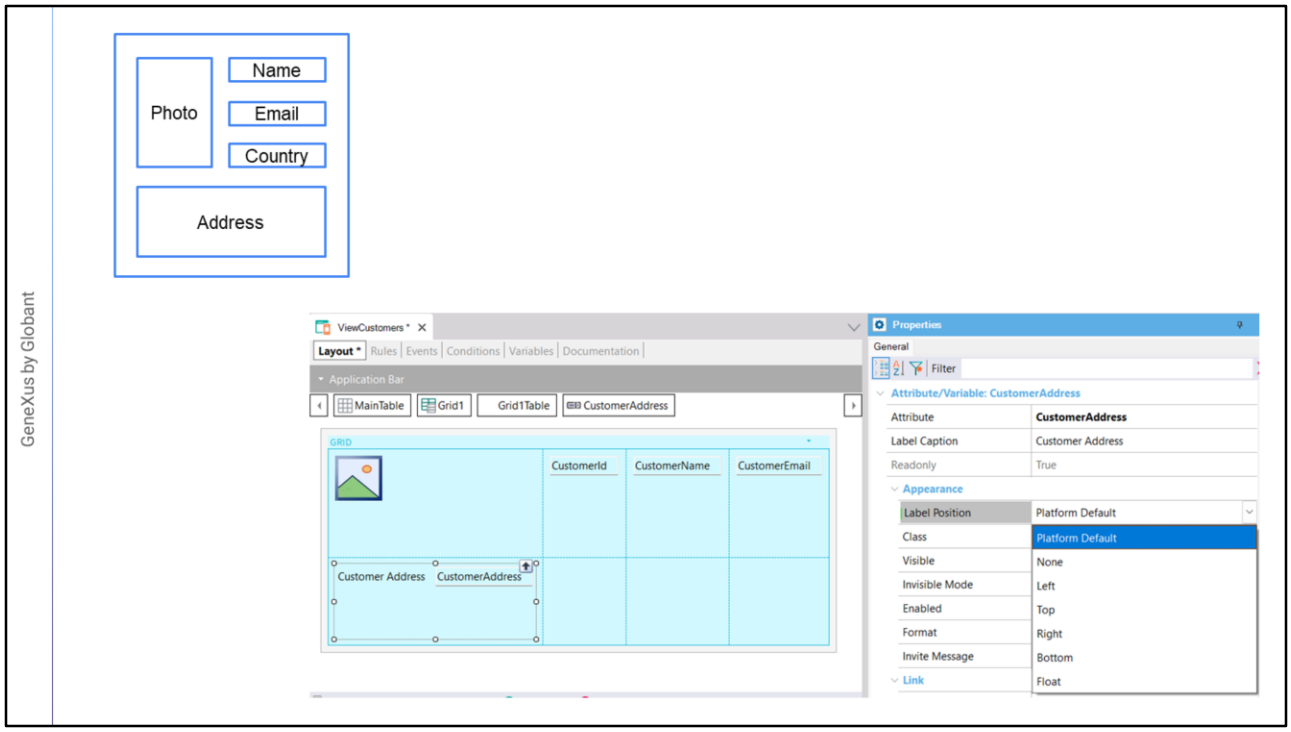
Ahora abrimos el objeto MainMenu y arrastramos el panel ViewCustomers al nodo Items y en la propiedad Image del action creado, le asignamos la imagen Customers.

Nota: para ver el menú tal cual se muestra aquí, crea el Panel de nombre ViewAttractions, arrástralo al nodo Items del objeto MainMenu y asigna en su propiedad Image la imagen Attractions. Trabajaremos sobre este Panel en un futuro video.

Recomendamos ejecutar la aplicación siempre con el emulador previamente abierto, para optimizar los tiempos de prototipado. Esto hará que cuando GeneXus quiera instanciar un emulador para ejecutar la aplicación mobile, encuentre este emulador en ejecución y lo utilice, con lo cual nos ahorraremos el tiempo de instanciación del dispositivo virtual y la ejecución será mucho más rápida. Ahora sí, presionamos F5.

Vemos que la lista de los clientes aparece sin ningún diseño, inclusive hay algunos datos que no llegamos a verlos por falta de espacio.

Trataremos de llegar al siguiente diseño tipo ficha, que nos parece el más adecuado. Cada fila del grid de clientes se mostrará con este diseño, a razón de un cliente por fila.



Notemos que el Id no se incluye en este diseño y que se desea mostrar la dirección del cliente que no habíamos incluido en el grid original. Vamos a agregar la dirección dando botón derecho sobre el grid y eligiendo "Agregar atributo o variable" y Customer Address.

Empecemos moviendo al atributo de la foto para el lado izquierdo del grid y arrastramos el campo dirección hacia debajo de la foto.

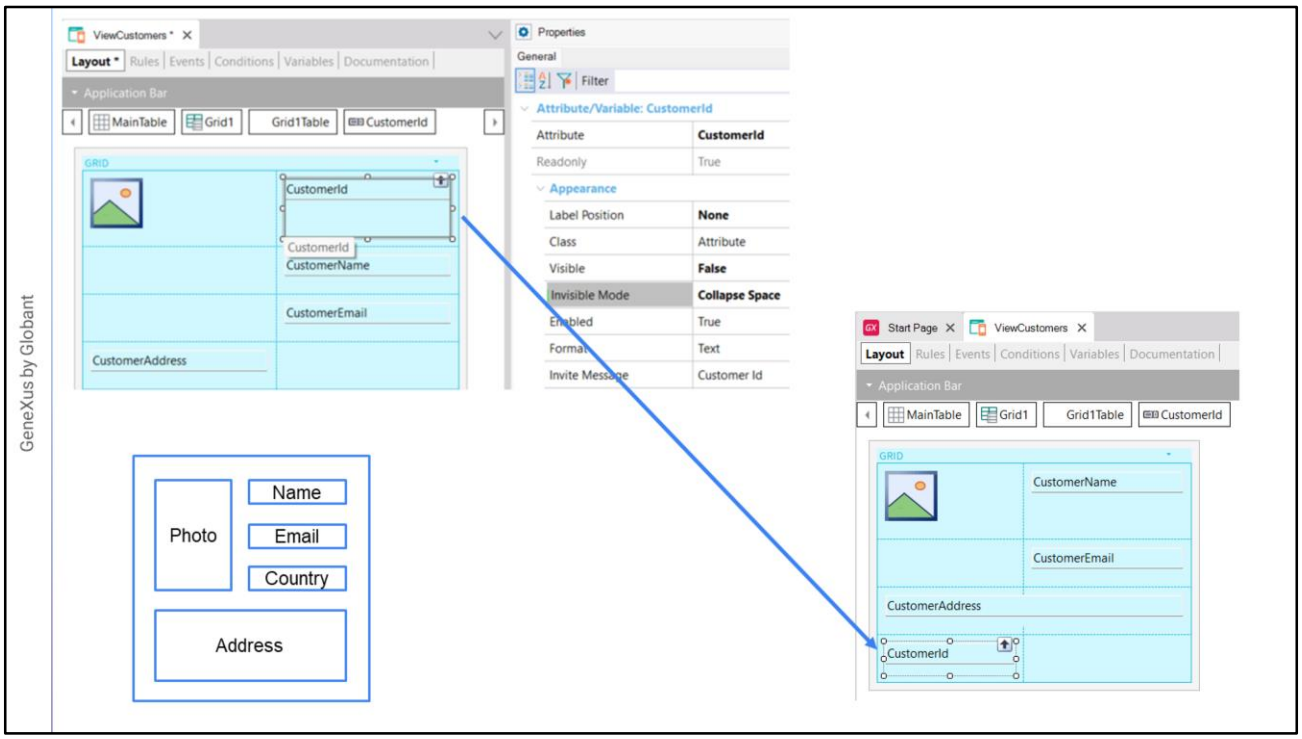
Vemos que CustomerAddress tiene una etiqueta a la izquierda. Si vamos a las propiedades del atributo, vemos que bajo la sección Appearence, la propiedad Label Position tiene el valor Platform Default. Para Android es el valor Top, arriba del contenido y para iOS es a la izquierda.

Si abrimos el combo vemos una serie de valores que nos cambiar el lugar por defecto donde se mostrará la etiqueta asociada al atributo.

El valor None significa que no se mostrará la etiqueta. Los valores Left, Top, Right y Bottom determinan que la etiqueta se mostrará a la izquierda, arriba, a la derecha o abajo del campo de contenido.

Con el valor Float la etiqueta se mostrará en la posición del contenido del atributo o

variable y cuando el usuario comienza a escribir en el campo, la etiqueta se moverá hacia arriba del mismo como si flotara. Esto solo tiene efecto en campos editables y en datos del tipo numérico o de tipo carácter, en el resto de los tipos se adopta el valor Top.



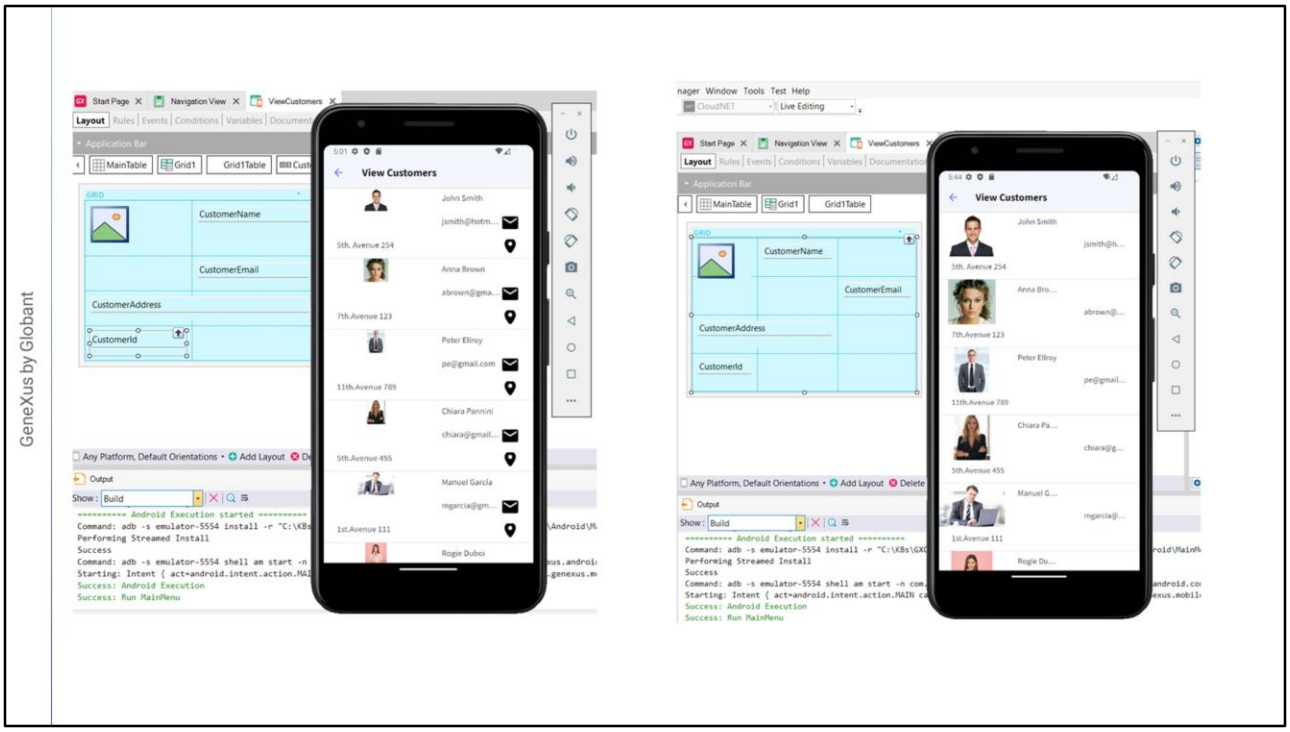
Para CustomerAddress vamos a elegir el valor None, con lo cual la etiqueta desaparece.

Movemos a los atributos de nombre, correo y país para debajo del CustomerId. Como no queremos mostrar el CustomerId, podemos quitarlo o poner su propiedad Visible en False. Si se puede necesitar obtener otros datos de un cliente seleccionado, debemos pasar por parámetro el valor de su identificador, por lo que deberíamos ocultar el atributo usando la propiedad Visible en False, en vez de borrarlo.

Debajo de la propiedad Visible está la propiedad Invisible Mode con el valor por defecto Keep Space y esto significa que aunque ocultemos el atributo como hicimos, se seguirá manteniendo el espacio que ocupa el mismo cuando se ve.

Si abrimos el combo vemos que también tenemos el valor Collapse Space, que significa que el atributo se ocultará y se colapsará el espacio, permitiendo que los otros controles ocupen el espacio disponible. Sin embargo, esta posibilidad solo funciona para contraer información de filas (colapso horizontal), pero no de columnas (colapso vertical), por lo que en nuestro caso como el CustomerId está en la misma fila que CustomerPhoto, debemos quitar el CustomerId de allí y si lo movemos para la última fila y allí sí se colapsará.

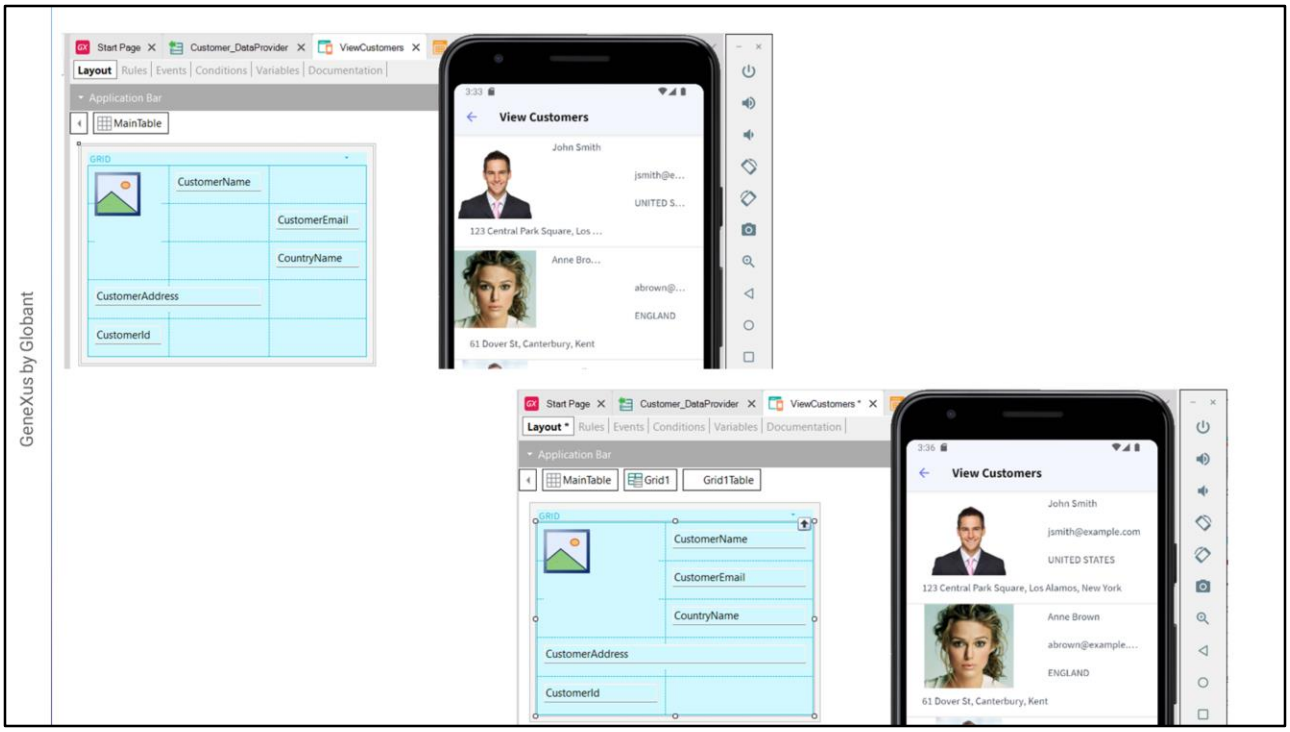
Vamos a acomodar un poco los controles para respetar el diseño al que queríamos llegar. Como necesitamos que la dirección ocupe el espacio de las dos columnas, vamos a la sección Cell Information y en la propiedad Col Span ponemos el valor 2.
Y presionamos F5.



Damos tap en el icono View Customers y vemos que se carga la lista de clientes.
Si bien se parece a lo que necesitamos, no se ve muy bien. Sí notamos que efectivamente no aparece CustomerId y que se colapsó el lugar que ocuparía.
Intentemos mejorar un poco el tamaño de la foto y hagamos que ocupe tres filas de alto, poniendo la propiedad Row Span en 3.

Antes de seguir con otros cambios, vamos a utilizar la funcionalidad del Live Editing que nos provee el IDE de GeneXus.

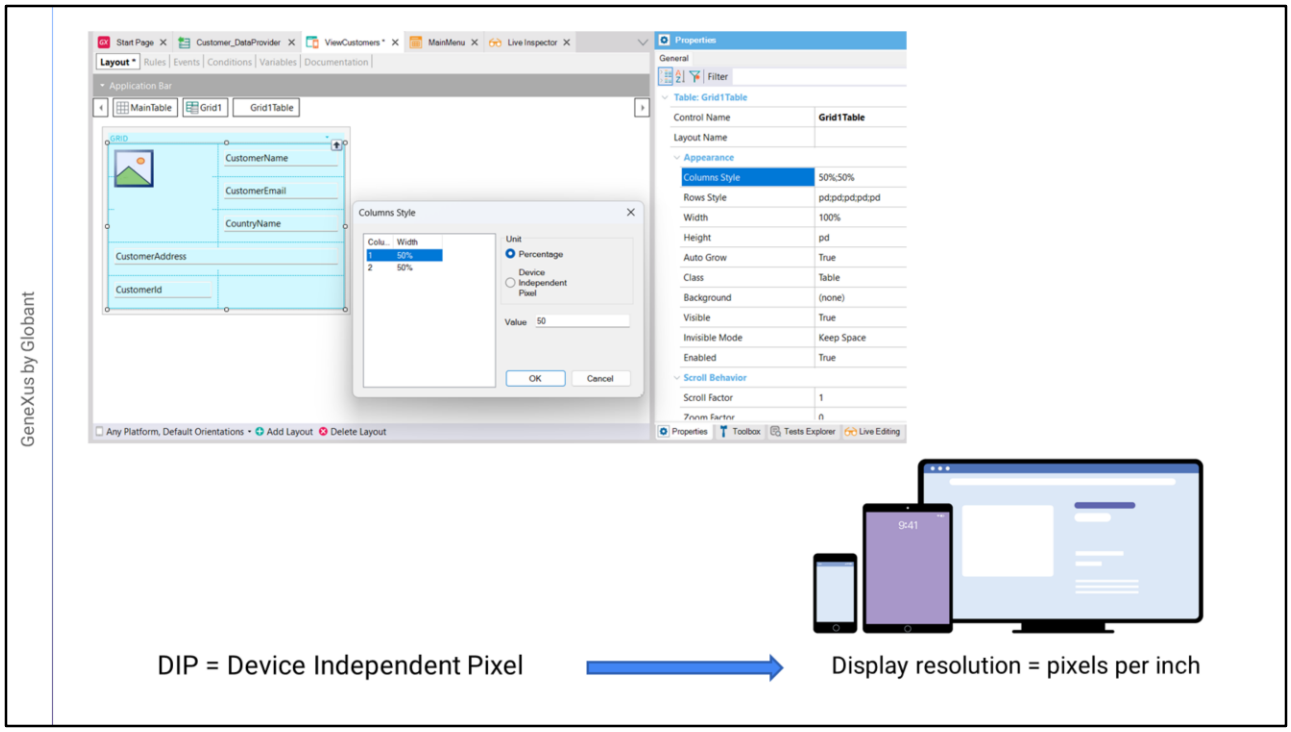
Para ver los cambios con el Live Editing, cambiamos en este combobox el valor Release por Live Editing y ejecutamos nuevamente la aplicación.
Comprobamos que el Live Editing está en funcionamiento viendo el tab Live Editing de la ventana de Properties, donde vemos que el IDE está conectado al emulador que teníamos en ejecución. También vemos que se abrió la ventana del Live Inspector, donde podemos identificar a los controles de pantalla y sus propiedades.



Volviendo a nuestro diseño, vemos que nos quedaron el nombre del cliente, su email y su país en diferentes columnas. Vamos a ponerlos en la misma columna y vemos que gracias al Live Editing, el cambio se reflejó inmediatamente en el emulador, sin tener que haber salvado o vuelto a ejecutar.

Esto nos permite hacer cambios en el diseño, como cambiar la posición, alineación, asignar otra clase, o cambiar las propiedades de una clase y así ir dejando la pantalla con una mejor apariencia viendo los cambios al instante.

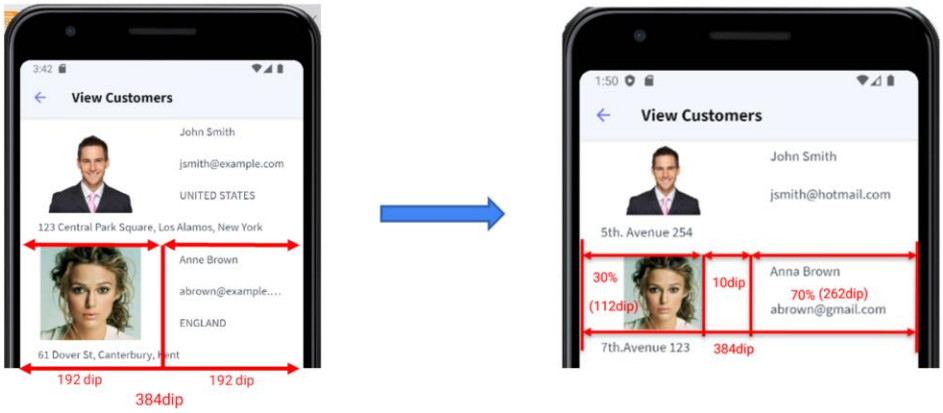
Vamos ahora a borrar la columna que está vacía, así que damos botón derecho y elegimos Delete Column. Vemos de nuevo que el cambio ya lo vemos en el emulador.



Para adecuar el tamaño de las columnas y filas, seleccionamos la tabla del grid, llamada Grid1Table. Bajo la sección Appearance, tenemos las propiedades Column Style, Row Style, Width, Heigh y AutoGrow que nos permiten cambiar el tamaño y comportamiento de los controles.

En ColumnStyle dice 50%;50% lo que significa que la tabla tiene dos columnas y que cada una está ocupando el 50% del ancho disponible. Abrimos esta propiedad y vemos a las dos columnas identificadas y que cada una tiene la unidad seleccionada en Percentage con un valor de 50.

Vemos que la otra unidad de medida son los DIPS: Device Independent Pixels. El Device Independent Pixel corresponde a una abstracción de un pixel que luego una aplicación convierte a pixeles físicos, lo que permite escalar a diferentes tamaños de pantalla. El dip para cada plataforma tiene diferente número de píxeles.

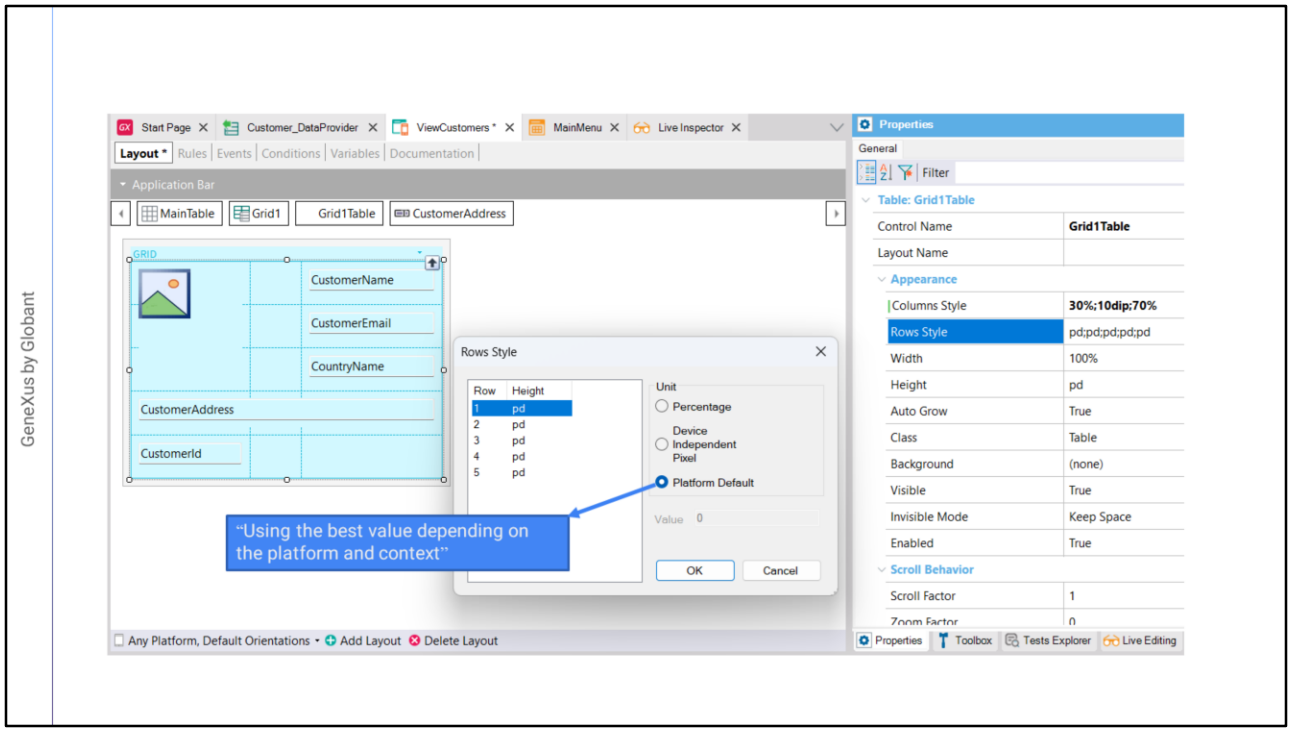


El ancho total es fijo y es de 384 dips, así que con este diseño cada columna ocupa el 50% o sea 192 dips.

Pero si hay alguna columna definida en dips, los valores en porcentaje son relativos al valor que resulta de restar del ancho total, los valores fijos (en dips). Por ejemplo si quisiéramos separar la columna de la foto de la de los textos, podemos agregar una columna como separador.

Si tuviéramos tres columnas, la primera con 30%, la segunda con valor fijo de 10 dips y la tercera con 70%, los valores que asumirán la primera y la tercera se obtienen de aplicar esos porcentajes al valor resultante de sustraer la suma de los valores fijos (aquí sólo uno, 10 dips de la columna separadora) del ancho de la tabla.

Debido a esto, la cantidad de espacio disponible para la primera y tercera columnas es de: 384 – 10 = 374 dips, por lo tanto la primer columna tendrá el 30% de 374 que equivale a 112 dips y la tercera el 70% restante, que son 262 dips.

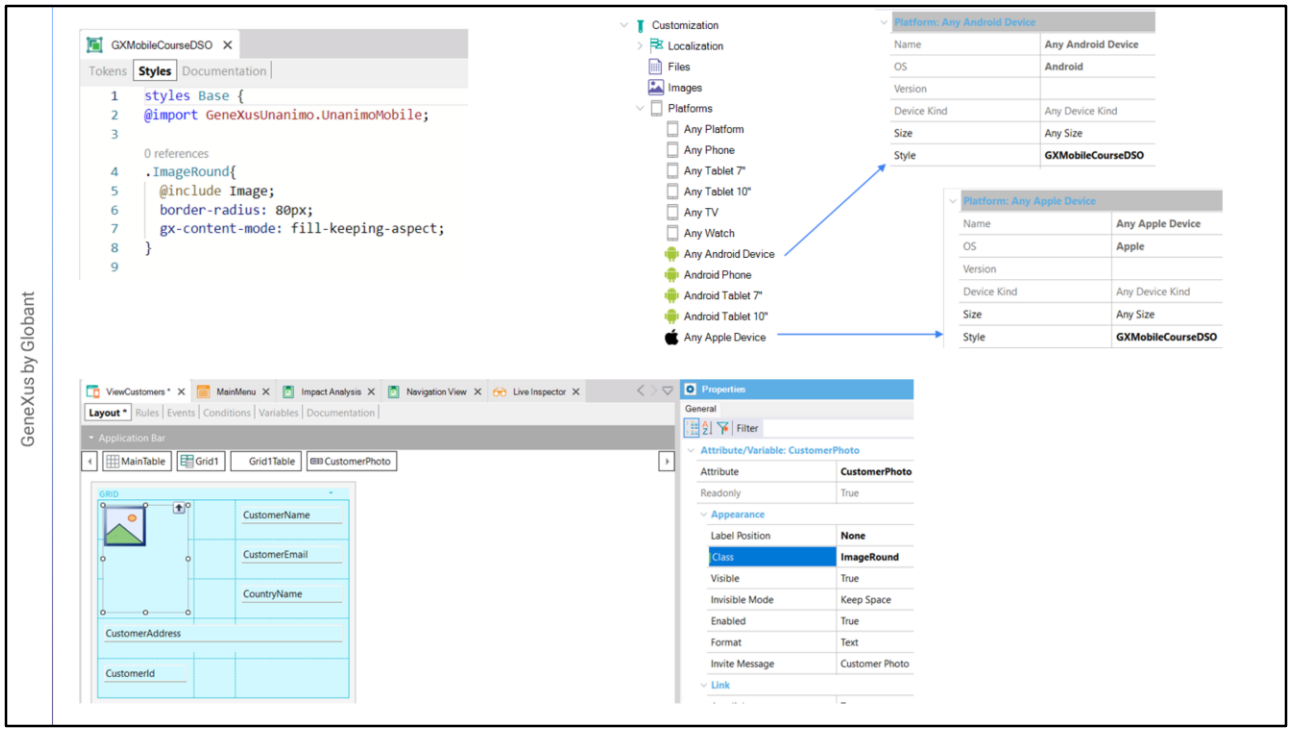


Vamos posicionarnos en la foto, damos botón derecho y elegimos Insert column after. Vamos a especificar los valores de Columns Style como habíamos visto, poniendo 30% para la primer columna, 10 dip para la segunda y 70% para la tercera, así vemos el resultado y gracias a que estamos usando Live Editing vemos que cambió y ahora tenemos un espacio entre el nombre y la imagen.

Si ahora vemos la propiedad Row Style, vemos que hay 5 filas con el valor pd: Platform Default.

Este valor difiere de plataforma en plataforma, y para una misma plataforma, también depende del contenido de la celda y si el campo tiene label o no, y si tiene, si la etiqueta será desplegada arriba o a la izquierda. El valor pd Corresponde a: "Usar el Mejor Valor Dependiendo de la Plataforma y el Contexto".

Por ejemplo, para Android, con Label Position = Top, corresponde a 64 dips, mientras que en iOS a 53 dips.



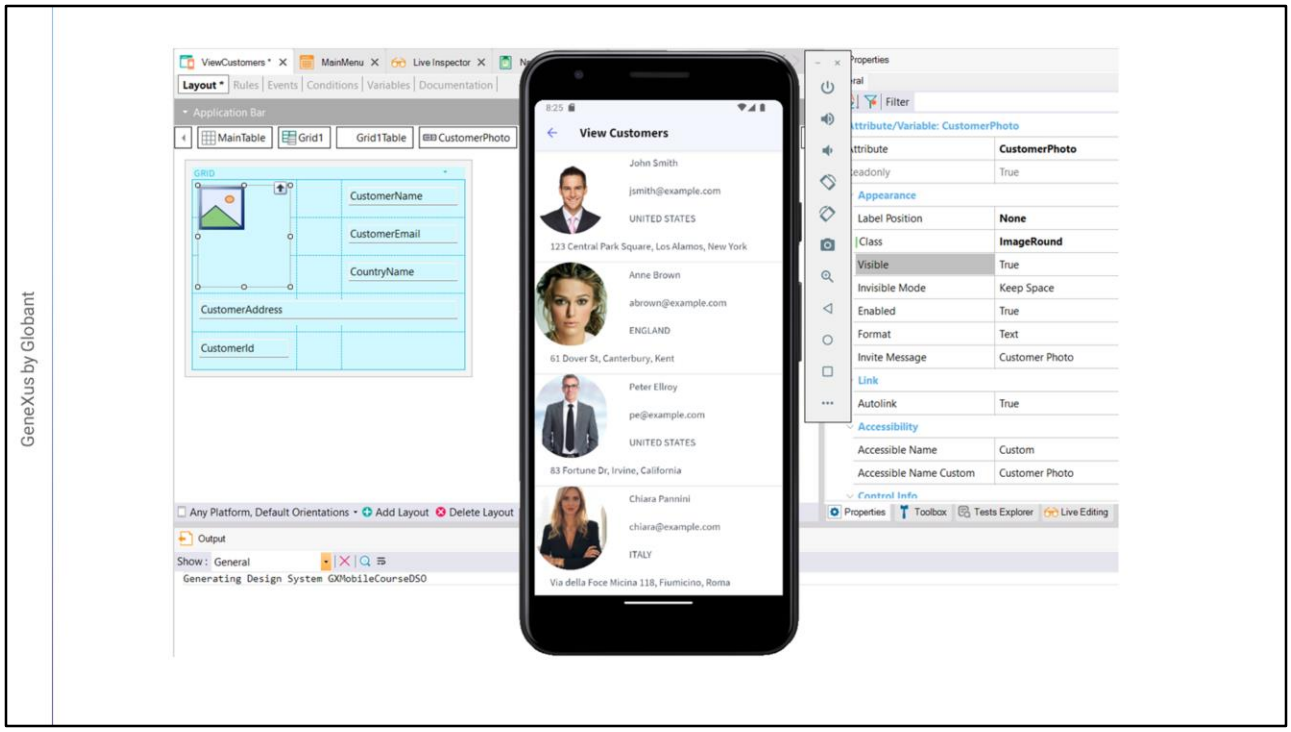
Revisamos la ejecución y las filas se ven bastante bien, así que dejamos estos valores.

Ahora vamos a cambiar la apariencia de la foto para que se vea redonda. Para eso vamos a asignarle al control de la foto una clase con propiedades que permitan esto.

Abrimos el objeto design system GXMobileCourseDSO que se importó con el xpz y en su solapa Style creamos una clase de nombre ImageRound. Le asignamos a la propiedad border-radius un valor de 80 pixels que hará que la imagen se vea redonda y a la propiedad gx-content-mode el valor fill-keeping-aspect con lo que la imagen se ajustará automáticamente al tamaño disponible sin cambiar su relación de aspecto, es decir, la proporción entre el ancho y la altura.

Para que este Design System Object sea el que se tome en cuenta en nuestros objetos mobile, debemos asignarlo a las plataformas en las que vamos a generar. As;i que abrimos el nodo Customization y luego seleccionamos AnyAndroidDevice en su propiedad Style le asignamos el design system GXMobileCourseDSO.
Lo mismo hacemos para la plataforma AnyAppleDevice.

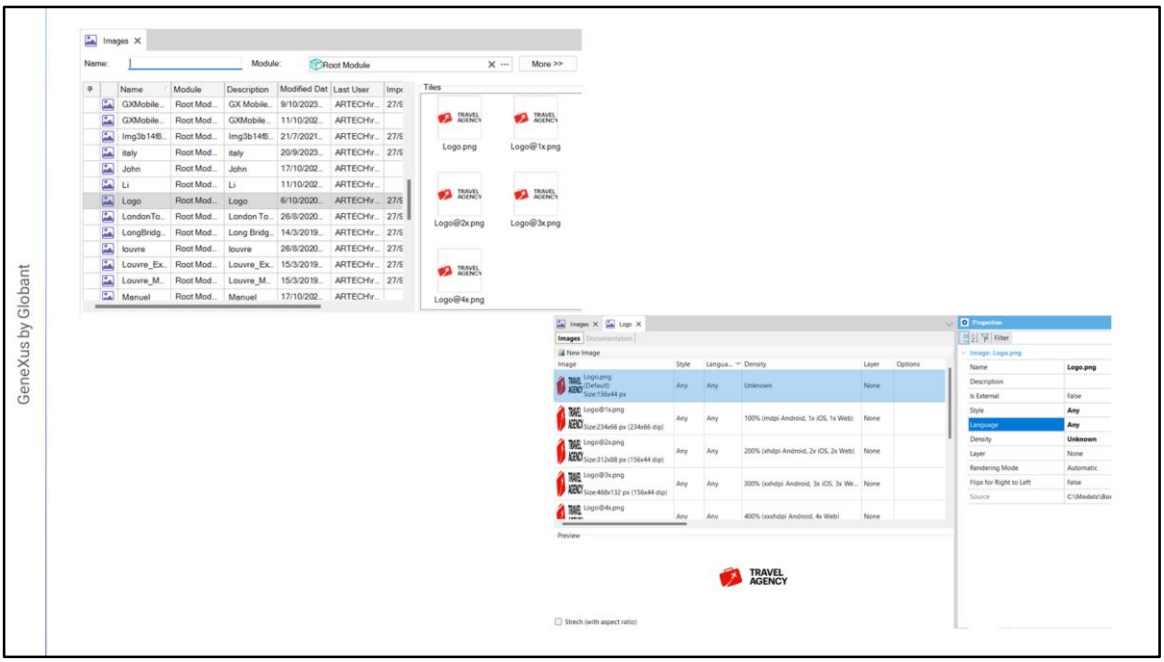
Ahora seleccionamos la foto y en su propiedad Class escribimos ImageRound.



Vemos que apenas asignamos la clase al atributo CustomerPhoto se ven los cambios gracias al Live Editing. Podríamos incluso cambiar los valores en la clase y también veríamos el cambio reflejado automáticamente sin ni siquiera salvar o compilar el objeto o ejecutar la aplicación. Esto favorece el desarrollo incremental.

En nuestro ejemplo estamos viendo la foto del cliente, que es una imagen almacenada en la base de datos, pero es frecuente que debamos usar una imagen fija, como un logo o una imagen de fondo.

En este caso, debemos tomar en cuenta que al agregar la imagen a la KB, debemos agregar en realidad varias imágenes con distintas resoluciones, ya que los dispositivos que ejecuten la aplicación pueden tener más o menos densidad y resolución de pantalla.



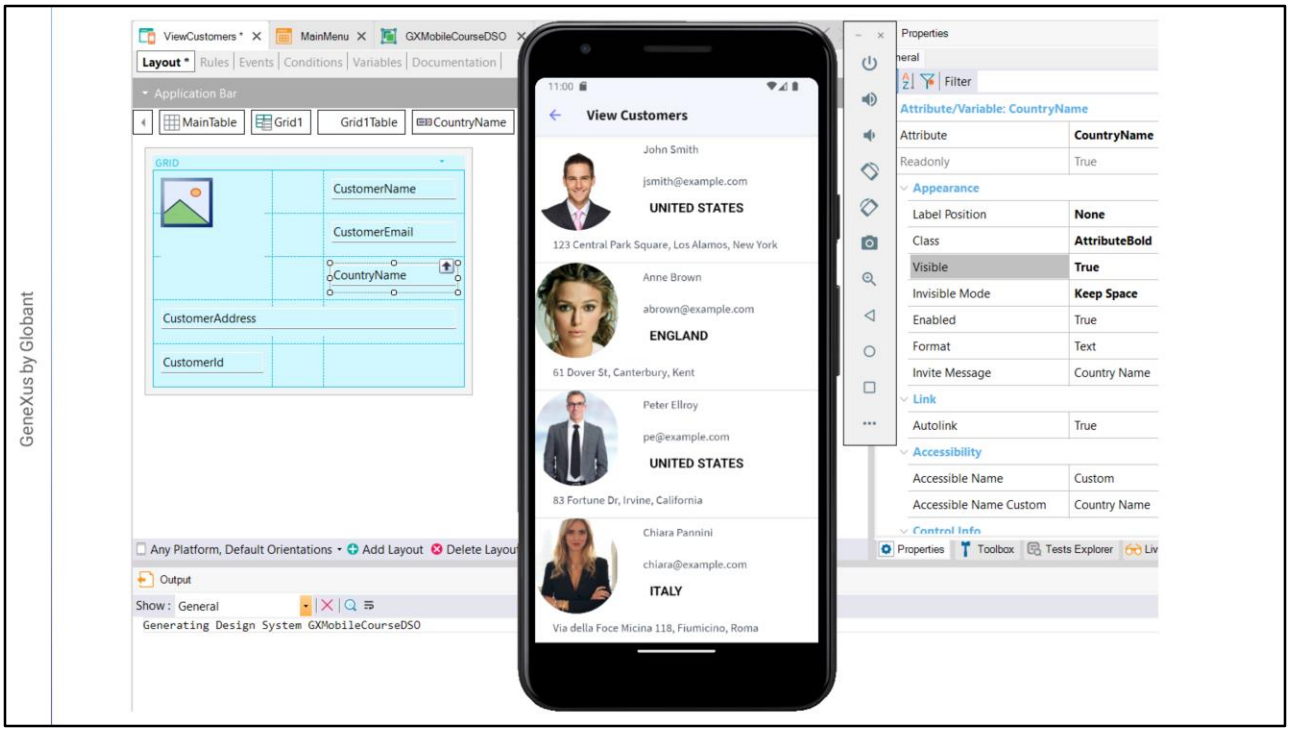
Si seleccionamos la imagen de nombre Logo, vemos que en realidad hay varias imágenes asociadas. Si abrimos la imagen, podemos apreciar que cada imagen tiene una resolución distinta y en el momento de ejecutar la aplicación se seleccionará automáticamente la adecuada dependiendo de la resolución de pantalla del dispositivo.

Además vemos que para cada imagen podemos especificar que estilo usará, es decir cuál es el Design System asociado. Esto permite poder subir imágenes diferentes asociadas a distintos Design System Objects. En nuestro ejemplo, el valor Any significa que la misma imagen se utilizará para todos los estilos definidos en la aplicación.

También podemos asociarle un lenguaje determinado, por ejemplo en el caso de que la imagen incluya un texto en su gráfico y que este texto sea distinto por idioma, de forma que podemos subir varias imágenes asociadas, una por cada idioma definido en nuestra aplicación.

Podemos subir imágenes con distinta densidad. Por ejemplo el valor de 100% corresponde a densidad media (aproximadamente 160 dpi) que se toma como línea de base tanto para Android como para Web y para iOS, en este último caso, el valor corresponde a pantallas que no son Retina Display. Recordemos que los dpi (device independent pixels) luego son convertidos a pixels reales de acuerdo a la resolución de cada dispositivo.

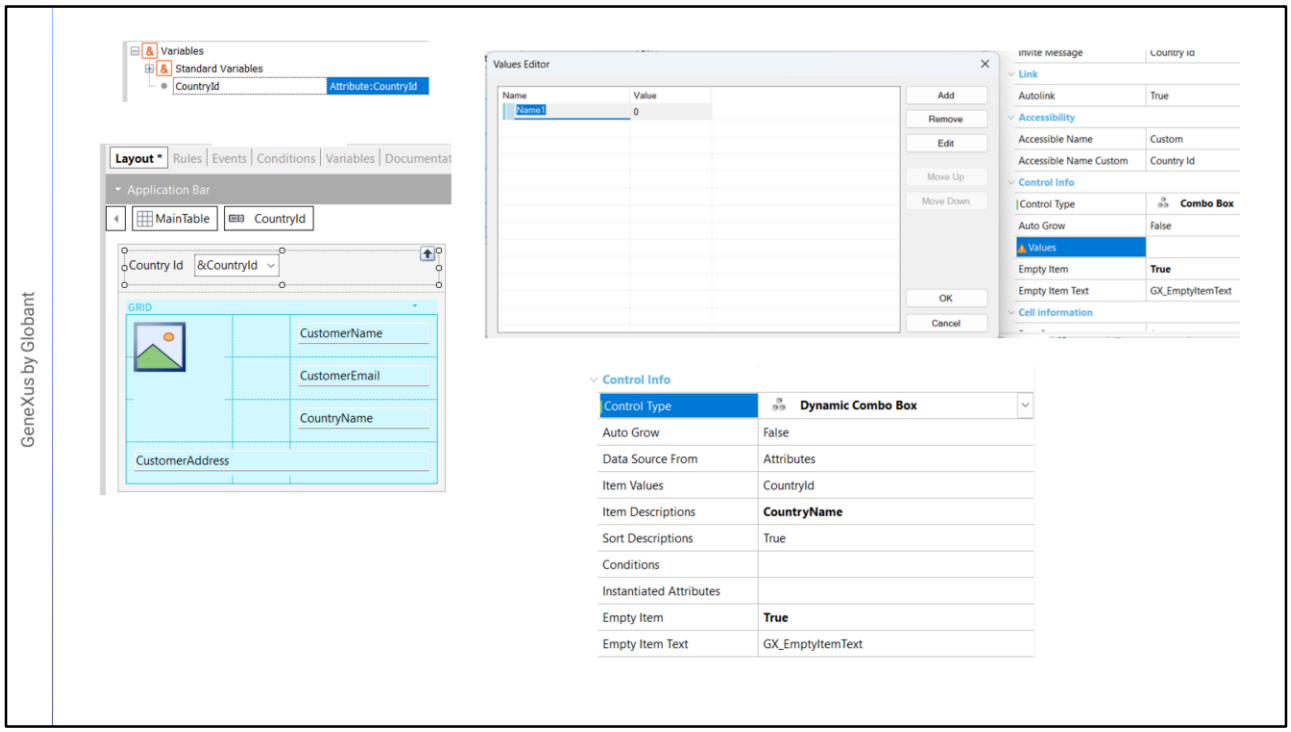
El valor 200% es para Extra-high-density (aproximadamente 320 dpi) correspondiente a imágenes Retina 2x para iOS y 2x de Android. El 300% corresponde a Extra-extra-high-density con unos 480 dpi, que son para Retina 3x de iOS y 3x de Android y el 400% corresponde a dispositivos con 640 dpis. En la medida que aparezcan dispositivos con valores mayores de resolución, este valor se irá incrementando.



Volvemos al desarrollo de nuestra aplicación.

Ahora vamos a hacer que el nombre del país salga en negrita, así que creamos una clase AttributeBold que herede de la clase Attribute, con las siguientes propiedades: font-weight en bold y la propiedad color en el valor: $colors.AttributeBolded que lo teníamos definido como token. Asignamos ahora la clase al nombre del país.

Aprovechamos y ajustamos todos los datos de la segunda columna con Horizontal Alignment en Left y Vertical Alignment en Middle. A la dirección le ponemos también alineación horizontal a la izquierda y la vertical en top.

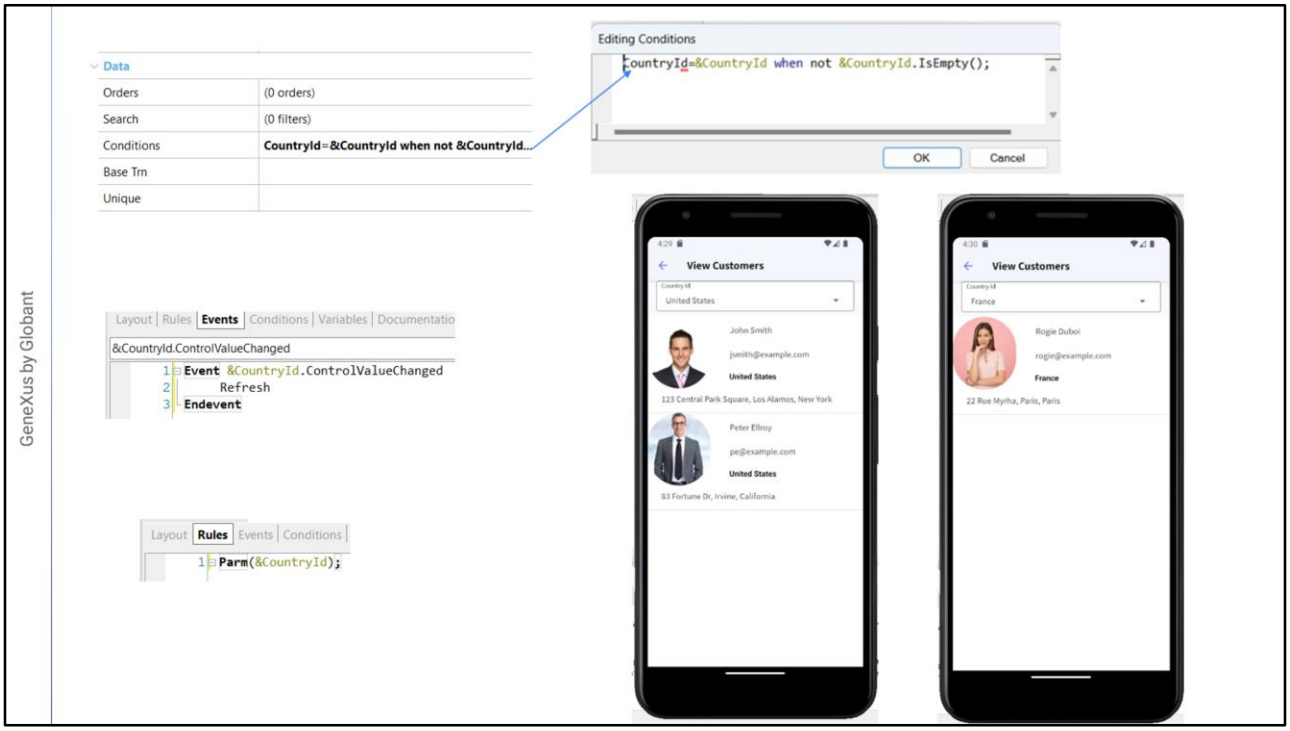


Muy bien. Para completar la funcionalidad de esta pantalla, vamos a agregar un combo box que nos permita seleccionar el país y filtrar los clientes por el país seleccionado. Para eso primero creamos la variable &CountryId que debido a su nombre, queda automáticamente basada en el atributo CountryId y la agregamos al form del panel, arriba del grid.

Si vamos a las propiedades de &CountryId, vemos que en ControlType dice Edit, por lo que por defecto esta variable se verá como un campo de edición. Para cambiar su apariencia y comportamiento, abrimos el combo de esta propiedad y vemos que podríamos asignarle el tipo ComboBox, pero en este caso deberíamos ingresar en la propiedad Value, los valores a mano para llenar el combo.

Pero nosotros ya tenemos la información de los países en la base de datos, por lo que en lugar de este tipo de combo elegimos un Dynamic Combo Box y vemos que en la propiedad Items Values queda asignado automáticamente con el atributo CountryId debido a que la variable se llamaba así, que será el valor devuelto por el combo al seleccionar un país.

El cuadro de propiedades nos indica que debemos asignar la propiedad Items Descriptions que será el campo cuyos valores aparecerán en el combo al abrirlo, así que asignamos al atributo CountryName a esta propiedad. Y por último, como queremos que el combo se inicie sin ningún país por defecto, ponemos la propiedad EmptyItem en True, dejando el valor del texto del ítem vacío por el asignado por defecto.



Para hacer que el país seleccionado en el combo actúe como filtro en la grilla de clientes, vamos a las propiedades del Grid y en Conditions agregamos: CountryId=&CountryId when not &CountryId.IsEmpty(); de modo que solamente se mostrarán aquellos clientes cuyo identificador de país coincida con el identificador del país seleccionado en el combo y en el caso de no haber seleccionado ninguno, no actuará el filtro y se mostrarán todos los clientes.

Ahora vamos a la solapa de eventos e insertamos un evento de la variable &CountryId, el ControlValueChanged y escribimos el comando Refresh. Esto hará que al terminar de seleccionar el combo se refresque el contenido del grid y se carguen los datos de la base de datos nuevamente, filtrados por el país seleccionado. Veremos más sobre eventos más adelante.

Por último, vamos a las reglas y agregamos una regla Parm con la variable &CountryId. Esto es necesario en esta arquitectura mobile debido a que para minimizar las consultas al servidor se prioriza el uso de datos cacheados y para que el servidor entienda que queremos traer nuevos datos, agregamos una regla Parm con las variables que usamos en los filtros, para que cuando cambien su valor se actualice la página.

Debido a que agregamos filtros y eventos que involucran al servidor, para ver los cambios tenemos que compilar nuevamente la aplicación, así que hacemos F5.

Abrimos View Customers y vemos el combo en la parte superior, que dice (None), para indicarnos que no hemos seleccionado un país todavía. Elegimos a Estados Unidos y vemos que el filtro funciona y se muestran únicamente a los clientes de ese país. Elegimos otro país y comprobamos que todo anda como queríamos.

En este video vimos cómo trabajar con controles de pantalla básicos, cómo mejorar su aspect y distribución en pantalla. En siguientes videos seguiremos viendo otros controles de pantalla con interesantes funcionalidades.